MSCA 31009
Machine Learning & Predictive Analytics
Joshua Goldberg
Thursday Session
October 5, 2018

Assignment 1 Regression (normal equation)

Use Linear equation normal equation to predict water temperature T_degCT
1) Only use 'Salnty', 'STheta' for predictors
2) Remove NaN / NA values from dataset (prior to building train/test sets).
3) Solve for rmse, variance explained, and r-squared.

Python

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression as lm
import sklearn.metrics as metrics

bottle_df = pd.read_csv("bottle.csv", low_memory=False)

bottle_df.dropna(subset=["Salnty", "STheta", "T_degC"], inplace=True)

X_train, X_test, y_train, y_test = train_test_split(
    bottle_df[["Salnty", "STheta"]],
    bottle_df["T_degC"],
    random_state=0)

X_train = X_train.assign(intercept=1)
X_test = X_test.assign(intercept=1)

""" Manual calculation """
theta_best = np.linalg.inv(X_train.T.dot(X_train)).dot(X_train.T).dot(
    y_train)
print("Coefficients: ", theta_best)

Coefficients: [-0.60203564 -2.62812934 99.08024365]

y_predict = X_test.dot(theta_best)

y_predict.head()

Out[9]:
844870    9.044284
734525   11.864221
687206    9.536117
470074    6.443909
275866    7.321033
dtype: float64
```

```
theta_best = np.linalg.inv(X_train.T.dot(X_train)).dot(X_train.T).dot(
    y_train)
print("Coefficients: ", theta_best)
y_predict = X_test.dot(theta_best)
Coefficients: [-0.60203564 -2.62812934 99.08024365]

lm_mod = lm().fit(X_train, y_train)
print('Coefficients: \n', lm_mod.coef_)

Coefficients:
 [-0.60203564 -2.62812934  0.        ]

y_predict_train_sk = pd.DataFrame(lm_mod.predict(X_train), columns=[
    "y_predict"])
y_predict_test_sk = pd.DataFrame(lm_mod.predict(X_test), columns=[
    "y_predict"])
""" Evaluate """
print("Model mean squared error: %.2f"
      % metrics.mean_squared_error(y_train, y_predict_train_sk.y_predict))
print("Model explained variance: %.2f"
      % metrics.explained_variance_score(y_test, y_predict_test_sk.y_predict)
         )
print("Model r-squared: %.2f" % metrics.r2_score(y_train,
    y_predict_train_sk))

print("Holdout mean squared error: %.2f"
      % metrics.mean_squared_error(y_test, y_predict_test_sk.y_predict))
print("Holdout explained variance: %.2f"
      % metrics.explained_variance_score(y_test, y_predict_test_sk.y_predict)
         )
print("Holdout r-squared: %.2f" % metrics.r2_score(y_test,
    y_predict_test_sk))

Model mean squared error: 6.50
Model explained variance: 0.64
Model r-squared: 0.64

Holdout mean squared error: 3.21
Holdout explained variance: 0.82
Holdout r-squared: 0.82
```